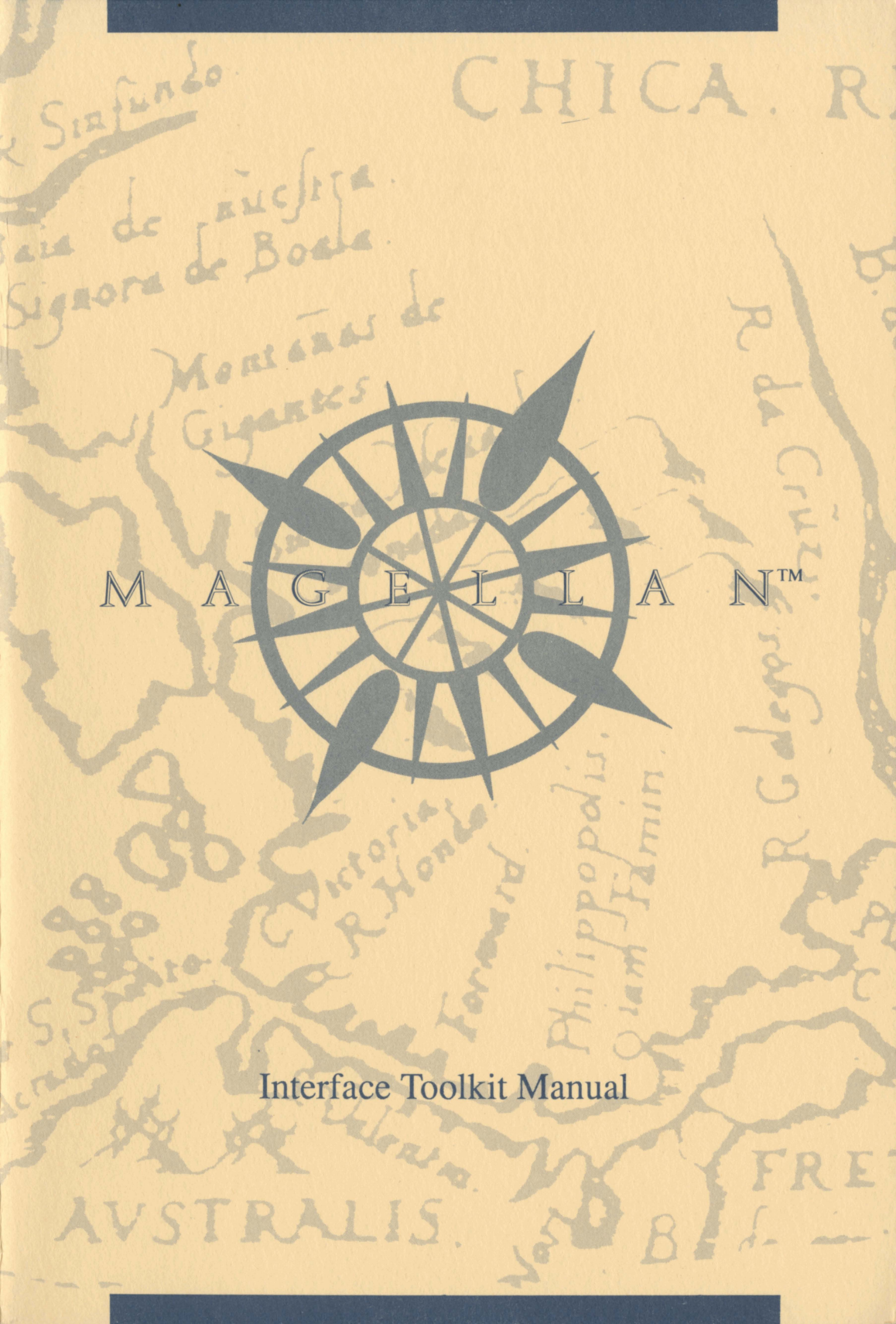

MAGELLAN™

Interface Toolkit Manual

Table of Contents

Preface

The Interface Toolkit is a group of programs designed to accompany the MAGELLAN™ Introductory Expert System Development Tool, Version 1.1 and user's manual.

The operators, GET and PUT have been added to MAGELLAN to provide an interface to the outside world, using programs like the modules in this Toolkit. The GET operator allows the inference engine to access information outside the expert system application or to retrieve information through the use of a specially designed interface. The PUT operator writes information from the knowledgebase environment to an existing file, port or device. These new operators, and the interface modules included in the Toolkit, make it possible for MAGELLAN expert systems to read from and write to other environments.

In addition to the interfaces supplied in the Toolkit, you can build your own interface to other applications. The use of the GET and PUT operators is not restricted to the interfaces included in this package.

The five interfaces supplied with the Interface Toolkit are:

The ASCIO module: an interface to text files.

The SBFIO module: an interface to Superbase™ data files.

The SERIO module: an interface to the serial port.

The WKSIO module: an interface to spreadsheets in Lotus 1-2-3™ format.

The PORTIO module: a group of ARexx™ macros that, combined with the MAGELLAN V1.1 public port, will allow you to run MAGELLAN as a slave process, either from the CLI or another software program supporting ARexx.

Each module in the Interface Toolkit is documented with an example, so that you can easily see how the interface works and try the example out for yourself. These examples will show you exactly how to set up the necessary parameters to use the interface and will also show you how MAGELLAN will respond to its use. Although some examples may seem simplistic, they were chosen for their ability to clearly demonstrate use of the interface.

A Technical Note on the Interface Toolkit Modules

The Interface Toolkit modules were constructed in C and are, for the most part, self contained, making reference only to routines within the module itself or to standard C and Amiga library functions. PORTIO and SERIO modules make use of the Amiga Exec and Intuition functions. MAGELLAN Version 1.1 and the Interface Toolkit are supported by the the Manx Aztec C68K v3.6 C language compiler.

MAGELLAN Version 1.1, via the Toolkit, allows access to outside resources by using a combination of a built-in macro language and parameter passing interface. Some interfaces can be customized by modifying the included C source code for those modules. Developers who are familiar with the C language can manually adapt almost any feature to the interface.

MAGELLAN can interact with interpreted environments or any language generating run-time files, if results are returned in the format required. This format is documented in this manual. Interface program modules do not need to be written in C to be used with the MAGELLAN tool.

Getting Started

Interface Toolkit Components

Your MAGELLAN Interface Toolkit includes:

- one Interface Toolkit diskette
- one Interface Toolkit manual
- a warranty card and license agreement

If any of these pieces are missing, or if the diskette envelope has been opened, please contact your dealership or vendor.

The GET and PUT Operators

The Interface Toolkit increases the scope of MAGELLAN expert system applications by enabling the inference engine to interact with the external environment. The Toolkit uses MAGELLAN's two new operators, GET and PUT, to provide five specific interfaces for MAGELLAN. Using the GET and PUT operators in rules within your knowledgebase lets you automate system inquiries during the inference process. Instead of questioning the user for information, a MAGELLAN expert system can now be instructed to search for specifically needed information in an external application and store it back into the knowledgebase. When a rule containing a GET or PUT operator is triggered during inference, MAGELLAN is prompted to connect with the specified file and either GET information from that file or PUT information derived during inference into that file.

The GET and GET # operators will call any program that is specified in the value box of a rule clause. GET or GET# sends this program the remaining arguments given in the rule as well as some additional information. The GET operator stores the result(s) from its interface with the exterior file into the list of known values that are associated with the object and attribute being tested. These new values are then available for use in inference. The GET # Operator is used in exactly the same way as the GET operator but specifies that the information being retrieved is a floating point numeric value.

The PUT and PUT # operators command MAGELLAN to write a derived value to an external file. You can either create your own command files or just output results of inference events into environments that have been designed to use this information.

In either case, the type of the final data returned to MAGELLAN is determined by the type of the object-attribute (cell) that is being filled with the operator's action. GET# and PUT # tell MAGELLAN, as rules are created, that the cell *to be created* in this rule clause is a numeric cell. If you are creating rules and using MAGELLAN's ability to create cells for you (the simplest method) then you should use GET# or PUT# to indicate a numeric cell, and use GET or PUT to indicate a text cell.

In short, the GET and GET# operator command MAGELLAN to retrieve information from an external location and the PUT and PUT# operators command MAGELLAN to write information to an external location.

The following is from the MAGELLAN v1.1 User Manual Addendum, and is useful in understanding the GET/PUT protocol underlying the use of the modules included with the Interface Toolkit. If you do not wish to examine these underlying concepts now, you may wish to refer back to this section after using the later portions of this manual with any of the included interfaces.

GET and PUT Operators

To be truly useful, an expert system sometimes requires data maintained in conventional software. A company may have thousands of hours of work in a conventional database, and may wish to hook an expert system to it. MAGELLAN permits access to external programs and data via the GET/PUT operators. Specialized programs can be created to retrieve information from virtually any program. Additionally, the MAGELLAN Interface Toolkit (another product from Emerald Intelligence) can be used to hook to LOTUS 1-2-3™ compatible spreadsheets (Analyze!™ and MaxiPlan™) and also to databases created with SuperBase™. The INTERFACE TOOLKIT provides these 'hooks' for MAGELLAN to use via the GET/PUT feature. Additionally, the INTERFACE TOOLKIT provides source code (in 'C') for some of the interfaces provided. This is a big help to users who wish to create their own interfaces to programs of their own.

Using GET/PUT

Without the INTERFACE TOOLKIT, MAGELLAN can still make effective use of the new GET/PUT features. What is required is more effort on the part of the user. Creating a rule that uses the GET function is quite simple. Imagine a file (parts.file) has been created that contains the following information about parts in inventory:

name	number	price	count	weight
WIDGET	2134567	1.25	320	12.3
SMERMAK	3456777	.75	2100	2.6

Obtaining this information, and using MAGELLAN to determine how many SMERMAKS are in inventory, or how to ship a WIDGET based on weight is not complicated. A single rule could be used to obtain the information from the file 'parts.file', using a stand-alone program (this is the extra effort part) called PARTIO that would open a file (parts.file), find a 'record' for a part by name (WIDGET) , and take the first, second, third or fourth character string in that file, based on whether it was looking for the PART NAME's number, price, count or weight respectively.

Here's the rule:

```
IF
      part name  is  WIDGET
and   part data  is    desired

THEN
 part number  GET    partio  parts.file  WIDGET    NUMBER
 part  price   GET#   partio  parts.file  WIDGET    PRICE
 part  count    GET#  partio  parts.file  WIDGET    COUNT
 part weight  GET#   partio  parts.file   WIDGET    WEIGHT
```

Note the GET, GET# operators. The four THEN part operators all do the same sort of thing: they run a program called PARTIO (that you have created.. again the extra effort part), sending it several arguments, including:

PARTS.FILE -a file name to read data from

WIDGET - the part name passed to partio

NUMBER,PRICE,COUNT or WEIGHT - an argument showing which data element: number, weight, price or count, to retrieve for MAGELLAN

Backward chaining on part weight, for example, would ask the user for part name and whether the part data is desired. Once MAGELLAN is given both the name and the correct answer to the part data desired question, the GET's and GET#'s would instantiate the value of the object attributes they are hooked to in the THEN part of the rule.

GET/PUT Protocol

The MAGELLAN INTERFACE TOOLKIT makes the following much easier, as source code is supplied to base your interface programs on. When acting on a GET, GET#, PUT or PUT# operator, MAGELLAN takes the value part of the rule clause containing the GET or PUT-type operator and sends several command line arguments to an external program.

For example, from the previous rule:

```
THEN
 part number  GET    [ PARTIO   parts.file   name  number ]
```

sends the bracketed text to the AMIGA CLI, along with some additional necessary terms:

```
 PARTIO  getfname.tmp  R  blah  parts.file  name  number
```

This is just as if you had typed it into the CLI.

Let's examine the command line arguments. The bracketed comments designate whether you must specify the argument or whether MAGELLAN will automatically supply it:

PARTIO - the name of the program to run (you must supply this)

getfname.tmp - a temporary file (MAGELLAN will supply this)

R - read mode (MAGELLAN will supply this)

blah - data to write (MAGELLAN will supply this - not used in GET's)

parts.file - 1st argument to PARTIO - a filename (you must supply this)

name - 2nd arg. to PARTIO - a part name to look up (you must supply this)

number - 3rd arg. to PARTIO -the part number is desired (you must supply this)

So, the protocol for a GET program is the following:

Using the standard command line arguments in 'C':

argv[0] -- the name of the interface program used to 'GET' the data
argv[1] -- temporary filename for sending data back to MAGELLAN
argv[2] -- "R" or "W" meaning that MAGELLAN has commanded a read or a write from the program
argv[3] -- data from MAGELLAN if this is a WRITE to the program
argv[4] to argv[9] additional parameters for the called program, such as filenames to open, search strings, graphics positions or anything else needed to be passed from MAGELLAN to the program.

In the value portion of a clause with a GET or PUT operator, several of these arguments can be provided:

IF
[get] [data] [is] [desired]
THEN
[data] [value] [GET] [argv[0] argv[4] argv[5] .... argv[n]]

When the CLI command is created by MAGELLAN to actually do a GET or PUT, MAGELLAN will automatically add argv[1], argv[2], and argv[3].

To do a GET: Your program needs to:

-- set the environment variable "GETIO" to "PENDING" (this may be done using the setenv() function in MANX AZTEC 'C')
-- open a file named with the string sent in argv[1]
-- interpret the later argv[]'s [4] to [9] if necessary, and perform any work
-- write your data into the file named in argv[1]
-- close the file named in argv[1]
-- set the environment variable "GETIO" to "COMPLETED"
-- return

To do a PUT: Your program needs to:

-- set the environment variable "GETIO" to "PENDING" (this may be done using the setenv() function in MANX AZTEC 'C')
-- use the data provided in argv[3], if desired.
-- interpret the later argv[]'s [4] to [9] if necessary, and perform any work
-- set the environment variable "GETIO" to "COMPLETED"
-- return

GET and GET#

The GET and GET # macro operators are used in a rule in the knowledgebase to retrieve information from an external file or program. GET is used to retrieve character strings and GET# is used to retrieve numeric values from the target file. You enter either operator into the operator box of a rule clause.

For Example

```
Rule#:[ 1 ]
 IF:
        [ Last   ][   Name   ] [ is ] [ Smith   ]
[AND] [ First   ][   Name   ] [ is ] [ Pete   ]
 THEN:
[user][city][GET][SBFIO city  users  lastname=Smith  firstname=Pete]
```

Here, the GET operator is used to extract the "city" of an individual from a Superbase™ database file, using the SBFIO module. As this rule is being built, providing the cells LAST NAME and USER CITY did not already exist, MAGELLAN would create both as TEXT type cells: LAST NAME because of the operator 'IS' in the first IF clause, and USER CITY because of the operator 'GET' in the THEN clause.

Note the spaces in the value box between "SBFIO", "CITY","USERS","LASTNAME=SMITH", and "FIRSTNAME=PETE". These spaces are significant and are used by MAGELLAN to separate the arguments to the called program.

Let's look at the protocol of the command line *entered into the value field* following the GET operator. This command line tells MAGELLAN where to find the information that it needs.

SBFIO city users lastname=Smith firstname=Pete

In this VALUE box of the THEN clause of this rule:

SBFIO is the name of the interface module. It can be the name of any of the modules included in the Interface Toolkit or an interface that you have created. In this case, SBFIO is the interface to Superbase files included in the Toolkit. The following arguments are specific to SBFIO, and may be different or unnecessary for other interface modules.

city is used by SBFIO as the search field, and its value is what you want to read into MAGELLAN. In this case, "city" is one of the fields in each record of the "users" database. This string must match the name of the field in the file exactly. SBFIO is not sensitive to the use of upper or lower case characters but any name that contains a whitespace character (space,tab, etc.) must be enclosed in quotes.

users is the name of the file where the desired information is located. In this case, users is the name of that database file that contains the field "city" that SBFIO is searching for and will return to MAGELLAN. Remember that the file specified may need to be in the same format as the interface module that you have specified. In this case, SBFIO (the SuperBase™ file reader) expects the data file to be a SuperBase file.

Some interface modules may also require file extensions to be specified with the file name:

lastname=Smith
firstname=Pete

These next two arguments specify the conditions that must be met in the target database file.

In this case, **SBFIO** must find the **"city"** of the user record in the file **"users"** with the **lastname = "Smith"** and **"firstname = Pete"**.

When this rule is triggered during inference, it will command SBFIO to retrieve the text string for "city" from the SuperBase file for the user PETE SMITH and return it to MAGELLAN (via GET). MAGELLAN will then store it in the set of values of the cell 'USER' 'CITY'.

If the GET operator does not return a value to MAGELLAN, then the value 'unknown' may be stored with the object-attribute pair. In some cases, MAGELLAN may ask the user for missing or unavailable data.

PUT and PUT

The PUT and PUT # operators may also be used in either the IF or THEN clauses of a rule.

```
  Rule #:[ 3 ]

 IF:
[spreadsheet][writing][ is ][desired]
 THEN:
   [ suggested ][ task ][ PUT # ][WKSIO   trans.wks   A4 ]
```

The PUT operator is used in this example to write a specific text string, (the value of the SUGGESTED TASK cell), into a spreadsheet file (TRANS.WKS) and a specific cell of the spreadsheet (A4).

The interface used with the PUT operator in this example is the WKSIO interface module, specifically designed to read and write information into Lotus 1-2-3 format spreadsheet files.

Generalized Format of the PUT Command Line

[PUT] [Module Name] [parameter(s)]

PUT - goes into the operator field of a rule

Module Name - specifies the interface module that you want to use. The example above called the WKSIO module.

Parameter(s) - additional specifics to guide MAGELLAN to the correct place in the file. In our example, MAGELLAN is being given the spreadsheet filename and cellname for cell 'A4' in this spreadsheet. Parameters will differ from module to module.

The ASCIO Interface Module

The ASCIO module is the interface designed to read information from and write information to ASCII text files. You can use the ASCIO interface module to access existing files written with your word processor or text editor as long as they are stored in as ASCII or "text" format. A simple format for a data file to read or write might be:

df0:PHONEBOOK --- a file containing the first name and phone number of the members of my local AMIGA user group.

```
TOM         313-555-9999
DICK        313-555-1111
GARY        313-555-1234
```

For example:

```
IF  [ action ][ type   ] [ is ] [ call GARY]
THEN
   [phone   ] [number ] [GET ] [ASCIO  df0:phonebook Gary]
```

This example tells MAGELLAN to use the ASCIO module and to look for a character string that matches "Gary" in the ASCII file "df0:phonebook". The information following this text string will be retrieved as the phone number. In this case, "313-555-1234" is retrieved and sent back to MAGELLAN, which writes it into the value of the cell "phone" "number".

ASCIO will retrieve the information in an ASCII file that immediately follows the text string that you specify, separated by a space. ASCIO is not sensitive to the use of upper and lower case or punctuation.

Format of the ASCIO GET Command Line

When ASCIO is used with the GET operator in a rule, you must use the following format in the VALUE part of the rule:

[GET] [ASCIO filename word]

GET or GET # - goes in the operator field in the rule. GET indicates this is a text cell and GET# indicates that this is a numeric cell.

ASCIO - goes in the value field, along with the rest of the command line, and specifies that in this case, it is the ASCIO interface program being used to retrieve (GET) the data.

filename - the name and path of the file which contains the ASCII text that you want to search.

word - defines the text string that appears in the ASCII file directly in front of the text that you want to retrieve, separated by a space.

Command line arguments must be separated with spaces in the value field of the rule clause. The simple implementation of ASCIO does not accomodate spaces in the search string (search word), but an improvement that could be made would be to allow any individual arguments containing a space to be surrounded by double quotes. Make sure that you put the entire command line, except the GET, into the value field of your rule.

Format of the ASCIO PUT Command Line

PUT, when used with ASCIO, writes ASCII format data into a text file following the search word that you specify, after a separating space.

For example:

IF
[phonebook] [update] [is] [desired]
THEN
[phone][number][is][313-555-6666]
[phone][number][PUT][ASCIO df0:phonebook Gary]

The THEN clause of this rule (if phonebook update *is* desired) will cause two effects:

(1) the cell 'phone' 'number' will be given the text value "313-555-6666"

(2) ASCIO will write the value of the cell 'phone"number' into the file "df0:phonebook", replacing the text (presumably a telephone number) that follows the word "Gary" in the file, separated from "Gary" with a space.

The general format used for ASCIO PUT is (the same as GET):

[PUT] [ASCIO filename word]

PUT or PUT # - goes in the operator field of the clause.

ASCIO - specifies the interface module that you are using.

filename - is the name and path of the file that you want to write data to.

word - the word that immediately precedes the data you wish to replace in the file, separated with a space.

Technical Notes on the ASCIO Module

Files that are to be parsed with the ASCIO module must contain ascii text only. ASCIO does not read or write graphics or numeric characters.

ASCIO can match character strings in formatted text, unless the formatting characters are imbedded in the text string in the file.

The general algorithm used in the ASCIO module is summarized as follows.

characters are read in from the file until a terminating character (a linefeed, end-of-file or space) is read

if the new string matches the string specified in the argument passed from MAGELLAN, then

if the mode is READ then

the next string up to another terminating character read in is written to the transition filename passed from MAGELLAN and the interface is complete.

if the mode is WRITE then

a. the character string passed in the command line from MAGELLAN is written to the target file at the point at the beginning of the next string following the matched string.

b. if the string to write is not the same length as the matching string from the file, the contents of the file are shuffled so that the new string will fit into the file and the integrity of the content will be preserved.

The contents of test files to be accessed by the present form of ASCIO yield the best results if entered in the following arrangement.

WORD DATA
WORD2 DATA2
etc.,

The PORTIO Module

The new public port installed in MAGELLAN V1.1 can receive and interpret ARexx messages as commands for some Magellan operations. The public message port added to MAGELLAN makes it possible for MAGELLAN to be run as a silent 'slave' process. Any program that can find the port "MAGELLAN.PORT" and send data to this port in the correct format can be a master to MAGELLAN. This includes all ARexx compatible programs and macros.

The possible functions that can be commanded in this way are:

1. loading MAGELLAN and a specified knowledgebase.
2. loading a set of previously saved values
3. clearing a set of inferred values.
4. initiating an inference Attempt on a previously defined set of goals.
5. Quitting MAGELLAN once inference is completed.

One method to command the port "MAGELLAN.PORT" is to use the message-passing capabilities of ARexx. Though not strictly necessary, the data structure of the messages sent to the port "MAGELLAN.PORT" coheres with the ARexx port message block structure. This makes it easy to use ARexx macros to command MAGELLAN in slave mode. The PORTIO module is simply a collection of a few of these macros, which can be used to stimulate MAGELLAN to perform operations in slave mode.

Sample macros included in the PORTIO module are:

1. Load-KBase
2. Load Values
3. Clear All
4. Attempt
5. Quit

MAGELLAN as ARexx Master

Under MAGELLAN versions 1.0 and 1.1, MAGELLAN could act as an ARexx 'master', commanding other programs via their ARexx ports. By creating a rule :

IF
ARexx activity is desired

THEN
[arexx][command] [$][REXX macroname]

the ARexx macro "macroname" would be sent to the program"rexx" and would be executed. This macro could be any kind of ARexx macro, and could cause any number of effects on the total multitasking environment.

Note that this would permit the execution of the macro "macroname", and possibly other parameters could be added to the value field, and sent off. This can be a very powerful mechanism for interprocess control.

Advanced Topics

The next step would be to tie the execution of Arexx macros to the result of GET/PUT. This would permit the user to create a rule of the type:

IF
ARexx activity is desired
THEN
[macro][name] [is] [start-it-all-up.rexx]
[macro][name] [PUT] [rexxio]

where:

macro name -- is a cell containing the name of the Arexx macro to be executed.

rexxio -- a standalone program that accepts the name of the macro to send to the Arexx-supplied program "rexx" and sends it there.

Rexxio has been included as an example of an extremely simple GET/PUT interface program. It has one argument, the name of the macro to run, and it simply executes "rexx" with that argument as a command .

The SBFIO Interface Module

The SBFIO interface supports search and modification functions of database fields in files with the SuperBase™ file format. This module can read text and numeric values from SuperBase format database files and can write text to SuperBase format database files. SBFIO does not write numeric values into fields saved with the numeric format.

For example:

IF
[user] [lastname] [GET] [SBFIO df1:regusers lastname usernumber = 0101]
AND [user] [lastname] [is] [Dubrowski]

THEN
[action] [ ] [PRINT] [Greeting.msg]

In this example:

SBFIO - the name of the module that you want to use.

df1:regusers - the name of the SuperBase file and its location.

lastname - the name of the field that you want to retrieve information from.

user number = 0101 - a parameter that specifies a condition to find the record you want to retrieve 'lastname' from.

Notes on the SBFIO Module

- SBFIO does not translate information from fields saved in the date format.

- Changes made to a SuperBase file using the PUT operator are restricted to data file changes only.

- Changes made to indexed fields will require you to re-index your database file when you next open it for use in SuperBase.

- The SBFIO interface does not support writing numeric or date values to similarly formatted fields within the data file.

- Be aware that when multiple fields within SuperBase are identically keyed, the SBFIO module may not be able to differentiate between them.

i.e. If there are several 'Smiths' in a file of phone numbers, SBFIO will test your parameters against the first 'Smith" only. If your parameter values do not match the ones in this record, SBFIO will not find the specified value, having done no further searches of the file.

For example: Smith Gary 123-4567
Smith Joe 234-5678

In this example, Joe Smith is contained in a separate record from Gary Smith. If Gary's record comes first in the file or if his name is indexed before Joe's, Joe Smith will never be found in a search on last name. In this case, you would need to do a sequential search of the database file to find information relating to Joe Smith.

Note also that some of the SBFIO interface is very simple and makes use of SuperBase™ Personal file format data. If advanced versions of SuperBase are incompatible with SuperBase Personal, or if the next revision of SuperBase fundamentally changes the structure of data, indices, etc., then SBFIO may not be very effective, and could possibly damage your files. In no case will Emerald Intelligence be responsible for the use of these interface modules, and this module in particular. SBFIO is meant as an example of how a database interface could be constructed. Use of these modules is therefore at your own risk to the databases, spreadsheets or files to which they are designed to interface.

The SERIO Interface Module

The serial port can be a valuable interface to external equipment, auxiliary terminals or other computers. The SERIO module permits MAGELLAN to command the reading or writing of data from or to the serial port. In the case of reading data from the serial port, SERIO is quite simple-minded; it reads until there is no more data to be read. In writing to the port it is even simpler: it sends the string to the port, and that's it. In simple cases, this is all that is desired: send a command to another piece of equipment, or wait for a data word from another terminal. To facilitate more advanced use of the serial port, we have included the source code of the SERIO module. It is very simple and does not perform extensive error checking or provide much other useful data.

Writing to the Serial Port (PUT)

Characters can be sent to the serial port in either ASCII or hexadecimal format. You can transfer hex characters to the serial port via the SERIO interface by entering a combination of hex character equivalents and markers at the command line in a knowledgebase rule clause. A string of hex characters may be needed to set certain flags or characteristics on a device.

For example:

IF
[transaction] [ ] [is] [confirmed]

THEN
[action] [ ] [PUT] [SERIO \FF\0C\0D]

The string \FF\0C\0D is translated into a three byte string of hexadecimal numbers, which is sent directly to the device at the other end of the serial port.

You also have the option of setting baud rate, parity bits and stop bits at the command line. This is helpful when dialing with modems and other devices which can handle character throughput at certain speeds.

For example:

IF
[transaction] [ ] [is] [confirmed]

THEN
[command][string] is [ATDT555-1234]
[command][string][PUT] [SERIO 1200 18 N 1]

This example assumes that a modem is the final receiver of the character string in the THEN-value part of this rule.

The command line 'SERIO 1200 8 N 1" initiates a call to the **SERIO** interface, opening the serial port at **1200** baud, **8** bits, **no** parity and **one** stop bit. After the port is open, SERIO sends thecontents of the MAGELLAN cell 'command' 'string' (which is "ATDT555-1234") through to the serial port.

ATDT is a command string that a Hayes compatible modem recognizes as a request to dial. The modem will dial **555-1234**, the numbers that are supplied in the remainder of the string.

Reading from the Serial Port (GET)

The SERIO interface does a limited read function which accepts one character from the port. You can alter this number as well as put in your own result handling code by changing the ReadSer procedure present in the SERIO code.

A command to read from the serial port might look like this:

IF
[Result] [Char] [GET] [SERIO "600 7 N 1"]
AND [Result] [Char] [is not] [unknown]

THEN
[transmit] [Char] [parse] [transmit = result]
[transmit] [Char] [put] [SERIO "600 7 N 1"]

What happens in this rule (which is simply an echo function) is the following:

• the IF clause #1 gets a character from the serial port by requesting a read of the port. In this case, the SERIO module is called again, opening the serial port at 600 baud, 7 bits, no parity and one stop bit.

• The GET command tells SERIO to wait until it gets a single character from the serial port before it returns.

• Once RESULT CHAR is set with a character via the MAGELLAN 'GET' function, another cell 'transmit' 'char' is set to the value of 'result' 'char'. This is done by setting a variable name (or an alias) for each cell in the object edit feature of MAGELLAN 1.1.

• So, if these variables have been set up (transmit is a variable to represent the cell 'transmit' 'char' and result is a variable to represent the cell 'result' 'char'), clause #1 of the THEN part will set 'transmit' 'char' to 'result' 'char'. Clause #2 then will send the received character back out the serial port, completing the echo.

Some factors to keep in mind with this echo function include:

(1) GET will currently timeout after a few seconds. If waiting for the serial port to respond, SERIO may cause GET to timeout, and simply abort the GET function. In future MAGELLAN upgrades, this timeout will be under user and system control, but is currently used as a safety net for faulty interface programs.

(2) Though fundmamentally recursive in nature, this rule may not be detected as recursive. If it is, it may not permit the assignment of the variable values to occur as stated. In short, MAGELLAN may say that this example won't fly becuase you are changing the value of 'result' 'char' in both the IF and the THEN parts of the rule.

The WKSIO Interface Module

The WKSIO interface module is designed to read information from or write information to spreadsheets that have been saved in Lotus 1-2-3 file format.

For example:

IF
[combined] [assets] [GET #] [WKSIO balance.wks Assets]
AND [combined] [assets] [>] [10,000]

THEN
[credit][application] is [Approved]
[credit][application] [PUT] [WKSIO balance.wks A4]

In this example, MAGELLAN is retrieving the value of a range name called "Assets"from a spreadsheet named "balance.wks" and comparing it to a set value of 10,000. If thevalue returned by WKSIO to MAGELLAN into the cell 'combined' 'assets' is greater than 10,000, MAGELLAN will then write the text word "Approved" (the contents of cell 'credit' 'application') into the same spreadsheet in spreadsheet cell A4.

The elements in the GET command line are:

GET - goes in the operator field

WKSIO - the module that you are using

Balance.wks - the name of the spreadsheet that you want to retrieve information from. Note the '.wks' suffix on the spreadsheet filename; this is required by WKSIO.

Assets - the key that will help MAGELLAN find the right cell. MAGELLAN will take the value in the cell immediately following the title 'Assets'.

The elements in the PUT command line string are:

PUT - goes in the operator box.

WKSIO - the name of the interface module you want to use.

balance.wks - the spreadsheet that you want to put information into. You can always designate the location of these files if they are not in your current directory.

Approved - is the value that you want MAGELLAN to write to the spreadsheet.

A4 - this tells MAGELLAN exactly where to place this value. Specifying the exact cell is another method of designating the location. This technique or the one described about in the GET command string can be used interchangably.

WKSIO can read and write several different forms of data, including:

- text contained in a cell, like labels
- numeric strings from specific cells, in the form of integers or floating point numbers.
- results of calculated formulae

In each of these three cases, you must specify they location of the information being retrieved, either as a cell coordinate or a cell or range name. Remember that you can use the GET # operator to specify that you are retrieving a numeric value.

Some restrictions apply when using the WKSIO module:

- actual formula variables cannot be GET or PUT.
- updating to the worksheet is limited. You must:
  - maintain cell type.
  - write only to cells that already exist, either containing data or not.
  - WKSIO cannot make structural changes to your spreadsheet.

Other Interfaces

The GET and PUT operators can use the modules included in the Interface Toolkit, but are not limited to these interface programs. You can also write your own interfaces to access other file formats, ports, or external devices.

Any interfaces that you develop for MAGELLAN must follow MAGELLAN's GET/PUT interface protocol. The output from your interface program must be written into the transition file specified by MAGELLAN and passes as an argument to your program when it is called to make a connection. Your interface does not need to be written in C, but must be able to accept arguments from MAGELLAN in this format. Remember to separate your arguments with a space in your command line and surround any filenames containing whitespaces with double quotes.

This is the protocol, and some of these parameters are provided in the value field of the rule clause. Some of them, such as TempFile, Direction, etc., are supplied by MAGELLAN, but you must accomodate them in your handling of command-line arguments.

[Module] [TempFile] [direction] [ASCII value to pass] [arguments]

Module - is the name and location of the interface module that you want to use.

Tempfile - is the file where MAGELLAN will store its result. This filename will be generated automatically by MAGELLAN.

Direction - is a single character, R for Read or W for Write, to specify whether MAGELLAN is GETting or PUTting. This parameter will be generated automatically by MAGELLAN.

ASCII value to pass - is used with the PUT operator and ignored with the GET operator. This is where you specify the value that you want MAGELLAN to PUT into an external file.

Arguments - are optional fields used to further specify the location of external files for your application. You can use up to 4 parameters to specify a location or none at all, depending on your application.

MAGELLAN will assign character strings to your command line arguments as follows:

argv(0)	module -- must be supplied
argv(1)	MAGELLAN tempfile -- MAGELLAN supplies this
argv(2)	direction -- MAGELLAN supplies this
argv(3)	ASCII value to pass for PUT operators -- MAGELLAN supplies this
argv(4) etc.	arguments for file-- optional, you supply if needed

Handshaking Between MAGELLAN and Interface Programs

There is a simple handshaking process that occurs between MAGELLAN and any interface module:

(1) MAGELLAN creates the command line, using the parameters supplied in the value field of the rule, and adding additional parameters such as:

- name of the tempfile for the application to return information
- 'R' or 'W' for a GET or a PUT respectively
- The value of the cell in the rule clause to write, if PUTting data

(2) MAGELLAN sets the environment variable GETIO = REQUESTED

(3) The application program is started by MAGELLAN.

(4) MAGELLAN enters a time out loop waiting for a response from the application-- namely it checks the environment variable GETIO to see if it is COMPLETED or ERROR. If no result occurs in several seconds, MAGELLAN assumes GET/PUT Failure and aborts the GET/PUT process.

(5) Upon startup, the application program checks if GETIO = REQUESTED, if so, it set GETIO = PENDING as it does its work.

(6) When the application has completed its work, including opening, writing to and closing the temporary file name sent by MAGELLAN (in the case of a GET), or it has opened the data file, written the necessary data and closed the data file (in the case of a PUT), then the application sets GETIO = COMPLETED and exits. If desired, the application may set GETIO = ERROR, but MAGELLAN may not respond to this data.

(7) MAGELLAN, upon seeing the GETIO = COMPLETED, either opens the transfer file and converts the data into the cell value data for the object attribute value obtained (in a GET), or it simply continues (in a PUT).

Conclusion

The MAGELLAN GET/PUT interface is extremely powerful and can provide necessary hooks to external software, data structures or devices. The protocol, while seemingly complex, is really rather simple and provides a very flexible and 'hands-off' bi-directional data transfer capability. We believe that the Interface Toolkit is meant as a start, and the example interfaces only as sketches to get you going. The protocol itself can be used an unlimited number of ways to hook MAGELLAN to almost any kind of event in the AMIGA. As always, please keep us informed of your feelings, comments and problems as they come up. We are very excited by the potential of the GET/PUT interface and we look forward to your valuable feedback.

Emerald Intelligence would like to thank the following people for their help in bringing the MAGELLAN Interface Toolkit to completion:

Developed by Heidi Fredrick, Glenn R. Golden and David M. Kennedy.

Based on the original design of David M. Kennedy.

Documentation written by Heidi Fredrick and David M. Kennedy.

Layout, design and desktop publishing by Deborah A. Kennedy.

Special thanks to Cathy Foreman, Karen Bryja, Don Crouse, Jerry Barton, Tim Nolan and Stephen Kennedy.

MAGELLAN is a registered trademark of Lotus Development Corp. licensed to Emerald Intelligence Inc.

SuperBase is a trademark of Precision Software.

Lotus 1-2-3 is a registered trademark of Lotus Development Corporation.

Arexx is a trademark of William Hawes.

Emerald
Intelligence
3915-A1 Research Park Drive, Ann Arbor, MI 48108.